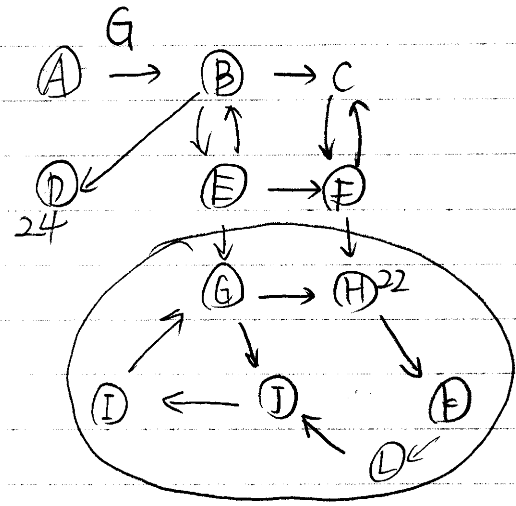
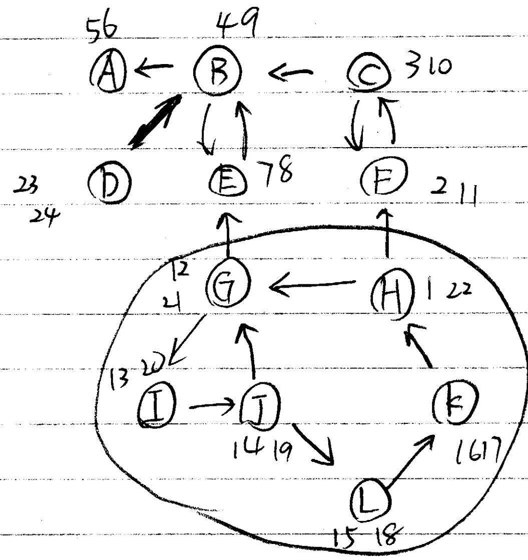
2018.3.6 Lecture 12.

A source vertex in a directed



sink scc



source scc

1) Do DFS numbering as reverse graph

2). Just traverse in decreasing order of post (V) in G and find all reachable vertice from V

every such series a scc

D 24 ↓ D

H ↓ HKLJIG

F ↘ FC

Lecture 12. Chapter 3.

3.4 Strong connected components

3. Strongly connected components (SCC)

Directed graph (with cycles)

SCC : maximal set of vertices that are mutually reachable from each ~~other~~

Connectivity in directed graph : $u \leftrightarrow v$.

$\{x, y\} \in SCC$

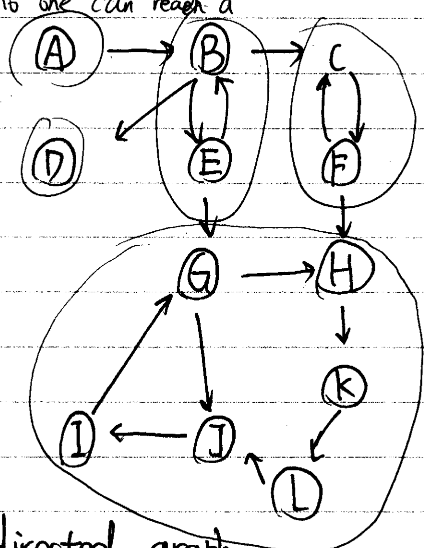
$\Rightarrow$ $\left.\begin{array}{l} x \text{ can reach } y \\ y \text{ can reach } x \end{array}\right\}$ mutually reachable

"largest" such set of vertice

↘ we can not add another vertice to SCC.

Directed graph has an ~~interrupt~~ DAG structure over SCC

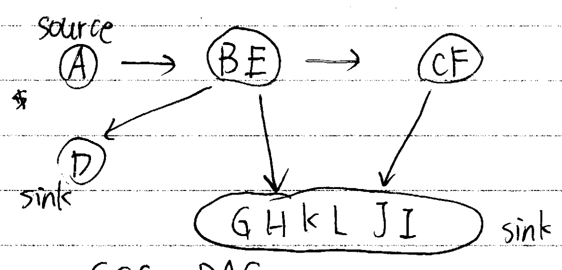
no one can reach a



directed graph

no one can reach A

D can reach nobody.



SCC DAG

Any directed graph is a DAG over ~~its~~ strongly connected component

SCC Algorithm

Given a directed graph $G = (V,$

1) Identity successively all "sink" nodes

2). From call sink , do a DFS to extrace the SCC remore it

→ remove it → mark all vertice as visited

Lecture 12

Topo sort:

1) Do a DFS numbering $O(|V|+|E|)$

2). ~~Sort~~ output vertice in decreasing order of post [V] value, order list

$\underbrace{\text{order of post [V] value}}_{\text{This is a topological sort}}$

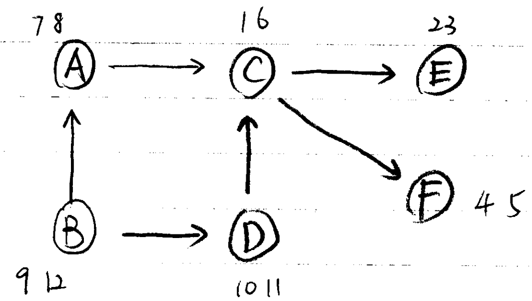
$\cancel{|V|\log|V|}$ $O(|V|)$

B (D A): C (F E)

12 11 8 6 4 3

$O(\cancel{|V|}+|E| + V\log(|V|))$

$O(|E| + \cancel{|V|}\log|V|)$ $\boxed{O(|V|+|E|)}$



post order list

E - 3

F - 4

C - 6

A - 8

D - 11

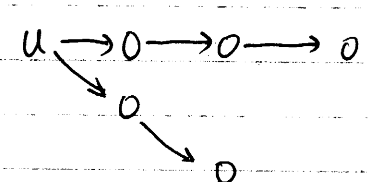
B - 12

sorting is for free

4. Proof of correctness

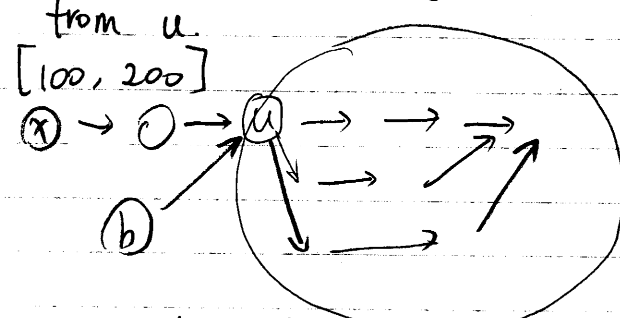
Interval of u [ ]

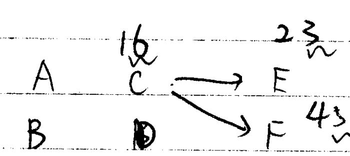
contains all reachable node's interval



post(u) must be larger than all other nodes that are reachable from u.

[100, 200]

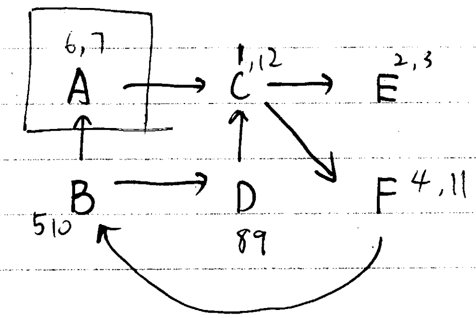


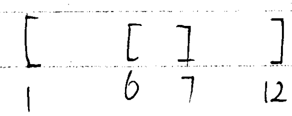


1) all descendants of u have smaller post value.

2). All ancestors of u have larger post value.

Lecture 12





Algo to detect cycles

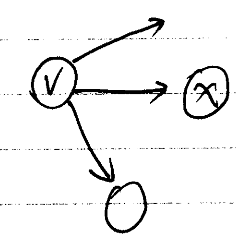
1) Do DFS intervals $\underbrace{O(|V| + |E|)}_{\text{linear time}}$

2) $\forall v \in V$

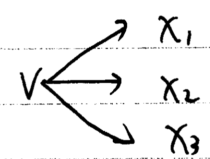
$O(|V|+|E|) \leftarrow$ for all $x \in$ neighbors of V

if interval[x] $\geq$ Interval [v]

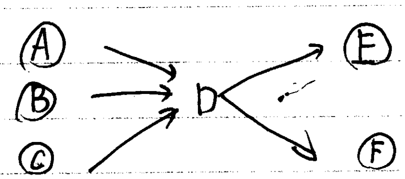
cycle found.



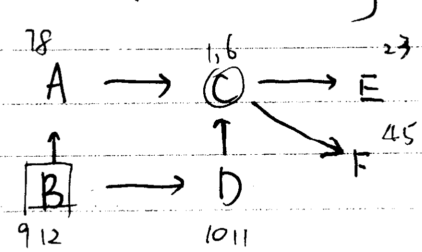
$O(|V| + |E|)$ worst case



3. Topological sort dependency graph
DAG



all of the "pre-requisits" of a node must before that node in ordering



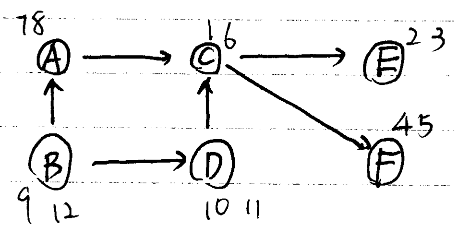
only possible for DAG.

2018.3.6

LECTURE 12

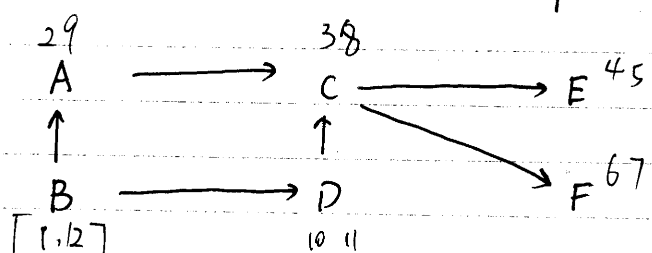
1. DFS numbering

pre-order number
post-order number



DFS numbering
many of them
1) Where we start
2) How we visit the outgoing edges

Containment of intervals ~~interests~~ indicates ancestor - descendent relationship



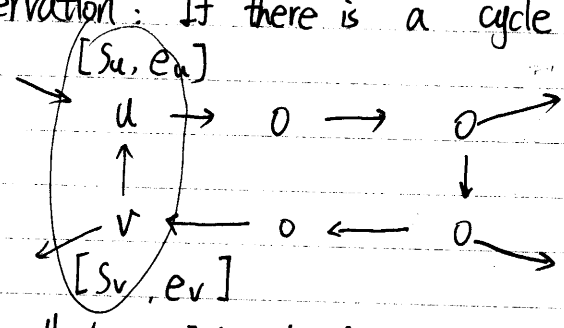
☆ If we start at a source then all vertices reachable from that source will have small interval

2. Detecting cycle

~~DAG → directed Acyclic Graph~~

give a directed graph, is it a DAG
→ Does it have a cycle or not ?

Observation: If there is a cycle, ∃ a node u and v such that



such that: Interval of v ≤ Interval of u

☆ $[S_v, e_v] \leq [S_u, e_u]$

If at v, we found a link to some vertex u such that Intervals of V ≤ Interval of u that has already been visited then ∃ a cycle

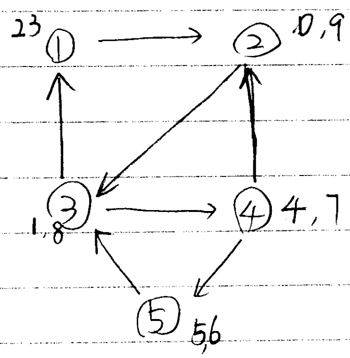
2018.2.27

Lecture 11 Chapter 3

1. DFS with pre & post number

order of visit in DFS exploration

directed graph



② [0, 9]

③ [1, 8]

① [2, 3] ④ [4, 7]

⑤ [5, 6]

DFS pre-order

DFS post-order

u. [x, y]

reachability can be checkes in constant time. after DFS.

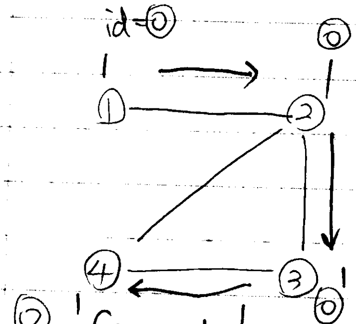
Q: is there a path from ② to ① ?

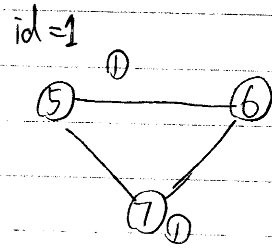
[0, 9] [2, 3]

since [2, 3] < [0, 9]

True !

Reachability: source encloses end.





disconnected graph

Connected components

↘ a maximal set of mutually reachable vertice

$CC_1 = \{1, 2, 3, 4\}$

$CC_2 = \{5, 6, 7\}$

DFS to find connectes components

CC(G)

for undirected

for all $u \in V$ | $O(n)$

  visited$(u) = 0$

id $= 0$

for all $u \in V$ ← $O(n)$

  if visited$(u) = 0$:

    DFS explore (u, id)

    id $=$ id$+1$

# of recursive calls is $O(n)$

cost of each call ?

DFS explore (u, id)

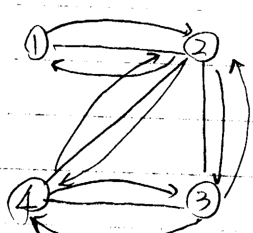
  visited$(u) = 1$

  id$(u) = $ id

  for all $x \in N(u)$ (neighbour)

    if visited$(x) = 0$:

      DFS explore (x)



every edge will be check exactly twice

$O(n)$ vertice

$+$

$2m = O(m)$

DFS $O(n+m)$

$\underbrace{O(|V| + |E|)}$

不管怎么走，都是每个 node 走两遍

for each $\{x, y\}$ once during explore x, once during explore y

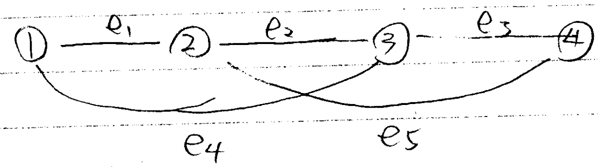
linear time algorithm in the graph "size"

Aggregate analysis : across all nodes

1). each node visites once

2). We look at all its neighbour.

2018.2.27 Lecture 11



Walk { ② — e_2 — ③ — e_4 — ① — e_1 — ② — e_5 — ④

② — ③ — ① — ② — ④

(2, 3, 1, 2, 4)

Path: ~~no~~ duplicate vertice allowed, except for the start & end

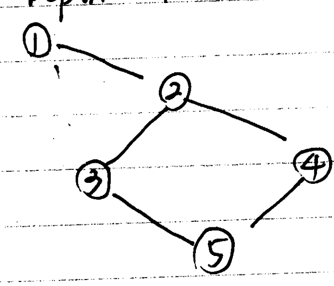
Cycle: is a path with same start & end

$\boxed{①}$ — ② — ③ — $\boxed{①}$

③ — ① — ② — ④

DFS: Depth First search

linear time procedure. (basic question: What part of graph are reachable fro a given vertex.



DFS explore (u) neighbour

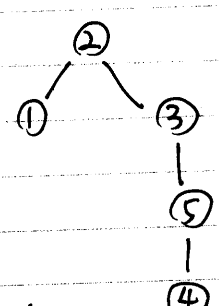
for $x \in N(u)$:

~~DFS explore (x)~~

if not visited (x),

DFS explore (x)

jump to neighbour with smallest ind



order of exploretion DFS tree

DFS → find all reachable vertices. from u

reachable: ∃ a path for u to s.

2018.2.27 Lecture 11

a graph is dense if $\underset{\uparrow \text{edge}}{m} = \Omega(n^2)$

2). Real-world : graphs are sparse.

$m = O(n)$

sparse ↑ a node / vertex connected to some max const # other nod

degree(v) = # of nodes that are adjacent to v

$= |N(v)|$ ← size of neighbour

eg: FB n = billion

u degree (u) ≤ 10^6 max

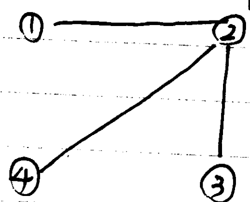
avg degree (n) ≤ 10 - 100

$O(n^2)$ edge ← max

For sparse graph

Adjacency lise representation

↳ keep track of presence of an edge.



array {
1 → 2
2 → 1 → 4 → 3
3 → 2
4 → 2

$|E| = |m|$

Q : (4, 3) exist?

(3, 4)

connect

~~O(1)~~

$O(|N(u)|)$

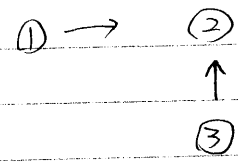
↑ space = $O(m)$ ← degree, edge

↘ $O(n)$ for sparse node

walk versuus path

walk is any sequence of alternating vertices & edge in a graph

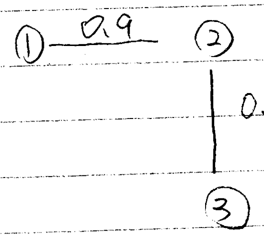
2018.2.27 Lecture 11



	V_1	V_2	V_3
V_1	0	1	0
V_2	0	0	0
V_3	0	1	0

directed

asymmetric matrix



	V_1	V_2	V_3
V_1	0	0.1	0
V_2	0.1	0	0.9
V_3	0	0.9	0

(Symmetric ?)

weighted adjcent matrix

Neighbour of a vertex V

$N(v) = \{u \mid (v,u) \in E\}$

set of all vertex "adjacent" to V

1						
2	1	0	1	0	0	1

$N(2) = \{1, 3, 6\}$

existence of edge $O(1)$

for adjacency matrix $N(v)$ takes $O(n)$ time

$A = n \times n$ matrix

$= O(n^2)$ quardratic space

has the record both the existence & absence of an edge.

1). Dense graph

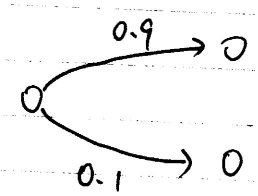
a large fraction of the possible edge exist

$G = (V, E)$

density $= \frac{|E|}{\# \text{ of possible edge}} = \frac{m}{\binom{n}{2}}$ ← ~~edge~~ # of actual edge ← # of possible edge

undirected $= |V| = n$ $\binom{n}{2} = \frac{n(n-1)}{2}$

Lecture 11



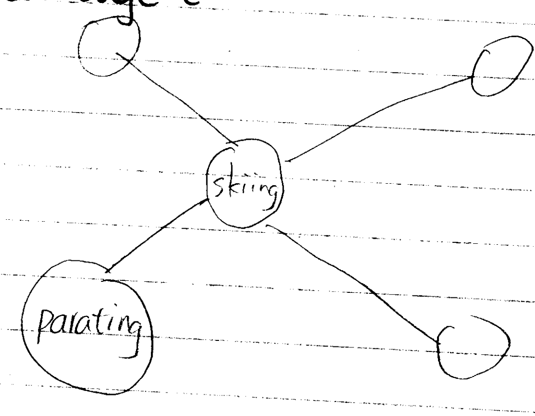
← probabilities graph

4. labeled vs unlabled
   ↳ there are "labels" on the verties & edge

$G = (V, E, L_V, L_E)$

$L_V(v)$ = label for v

$L_e(e)$ = label on an edge e



5. Representation

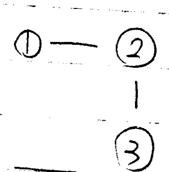
Adjecency matrix

$G = (V, E)$

$|V| = n$

$|E| = m$

$$A \quad \begin{array}{c|cccccc} & V_1 & V_2 & \cdots & & & V_n \\ \hline V_1 & 0 & 1 & & & & \\ V_2 & 1 & 0 & & & & \\ \vdots & & & 0 & & & \\ & & & & 0 & & \\ & & & & & 0 & \\ V_n & & & & & & 0 \end{array}$$



$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 3 & 0 & 1 & 0 \end{array}$$

$n \times n$ symmetric binary matrix

2018.2.27

Lecture 11

Depth first search

1. $G = (V, E)$

G ↑ graph V ↑ vertex $= \{v_1, v_2, v_3, \cdots\}$

$|V| = n$ order of graph

$E = \{(u,v) \mid u, v \in V\}$

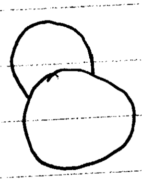
set of edge

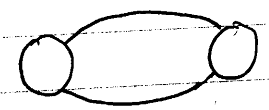
orderd pair ⓤ ⟶ ⓥ ← directed (u,v) (v,u)

unordered ⓤ ⟶ ⓥ ← undirected $(u,v) = (v,u)$

$|E| = m$ size of graph

2.

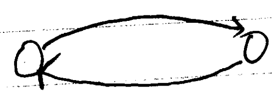




×

self - loop ×

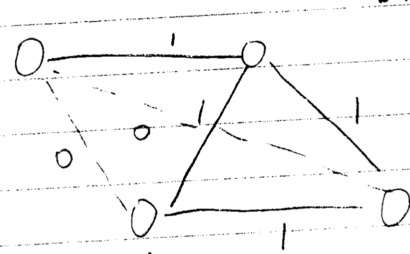
simple graphs ⇒ no self loop , no multigraph



✓

3. $G = (V, E)$

binary representation



weighted graph :

$G = (V, E, w)$

V = vertices

E = edges

$w(e) \in \mathbb{R}$ is the weight on edge $e = (u,v)$